Greedy Algorithms

เป็นขั้นตอนวิธีการแก้ปัญหาที่คิดง่าย ๆ แบบตรงไปตรงมา โดยพิจารณาว่าในขณะนั้นมีทางเลือกใดที่ให้ผลตอบแทนคุ้มที่สุด ขั้นตอนวิธีจะหาวิธีที่ดีที่สุดในขณะนั้น ถ้าข้อมูลในขณะนั้นพอเพียงก็จะให้คำตอบที่ดีที่สุดโดยทั่วไป Greedy algorithms ใช้กับปัญหาประเภท optimization ซึ่งเป็นการหาวิธีการแก้ปัญหาที่ดีที่สุด จากวิธีการแก้ปัญหาที่เป็นไปได้ทั้งหมด มักประกอบด้วยหลายขั้นตอน และแต่ละขั้นตอนจะต้องมีการตัดสินใจบางอย่าง

ดังนั้น Greedy algorithm คือ ประเภทของอัลกอริทึมที่มีแนวคิดว่าในแต่ละขั้นตอน จะตัดสินใจเลือกสิ่งที่ *ดูเหมือนเป็นสิ่งที่ดีที่สุด* ในตอนนั้น ๆ เสมอ

สำหรับบางปัญหา Greedy Algorithms จะให้คำตอบสุดท้ายที่ดีที่สุดจริง ๆ (Globally optimal solution) แต่หลาย ๆ ปัญหา การใช้ Greedy Algorithms จะไม่ได้ คำตอบที่ดีที่สุด เพราะฉะนั้น เมื่อเจอปัญหาใหม่ ๆ ขั้นแรกควรลองดูว่าสามารถใช้ หลักการ Greedy ในการแก้ปัญหาได้เลยหรือไม่ และได้คำตอบที่ Optimal หรือไม่ (อาจจะต้องพิสูจน์) (a) ถ้าได้ ก็ใช้เลย (b) ถ้าไม่ได้ ค่อยลองอัลกอริทึมประเภทที่ซับซ้อนกว่าแทน

ข้อดี	ข้อเสีย
– เขียนโปรแกรมได้ง่าย – Time Complexity ของโปรแกรมมักจะต่ำ เนื่องจาก ในแต่ละขั้นตอนการตัดสินใจเลือก สิ่งที่เลือกไปแล้ว มักจะไม่กลับมาเปลี่ยนการตัดสินใจทีหลัง และการเลือกสิ่งที่ดีที่สุดในขณะหนึ่ง ๆ มักทำได้ง่าย	ในหลาย ๆ ปัญหา คำตอบของ greedy algorithm ไม่เป็น global optimal solution

ปัญหาการเลือกกิจกรรม (Activity Selection Problem)

สมมติว่าเรามีห้องหนึ่งห้อง มีกิจกรรมที่จะต้องใช้ห้องหลายกิจกรรม แต่ละกิจกรรมจะมีเวลาเริ่มต้น และเวลาสิ้นสุด ต้องการเลือกกิจกรรมให้ได้มากที่สุด โดยแต่ละกิจกรรมที่ถูกเลือก ต้องมีเวลาไม่ทับซ้อนกัน ยกเว้นเวลาสิ้นสุดและเวลาเริ่มต้นของกิจกรรมต่อไป สามารถเป็นเวลาเดียวกันได้ เช่น มีกิจกรรม 11 กิจกรรม เวลาเริ่มต้นและสิ้นสุดของกิจกรรมที่ i แทนด้วย s_i และ f_i ตามลำดับ ดังนี้

i	s_i	f_i
1	0	6
2	1	4
3	2	13
4	3	5
5	3	8
6	5	7
7	5	9
8	6	10
9	8	11
10	8	12
11	12	14

ปัญหานี้สามารถใช้ greedy algorithms ได้ โดยเลือกกิจกรรมที่มีเวลาสิ้นสุด (f_i) น้อยที่สุดก่อน เป็นตัวเลือกที่ดีที่สุดในขณะนั้น เพื่อที่จะได้มีเวลาเหลือมากที่สุดจะได้เลือกกิจรรมที่เหลือได้มากที่สุด จากนั้นพิจารณากิจกรรมที่เหลือ เลือกกิจกรรมที่มีเวลาเริ่มต้นไม่ซ้อนกับกิจกรรมที่เลือกไปแล้วและมีเวลาสิ้นสุดน้อยที่สุด ทำเช่นนี้ไปเรื่อยๆ จนกว่าไม่สามารถเลือกกิจกรรมใดได้อีก สุดท้ายกิจกรรมที่ถูกเลือกจะมีจำนวนมากที่สุดเท่าที่จะเป็นไปได้

เขียนเป็นขั้นตอนวิธีได้ ดังนี้

1. เรียงกิจกรรมตามเวลาสิ้นสุด (f_i) จากน้อยไปมาก
2. ให้ n แทนจำนวนกิจกรรมทั้งหมด
3. เลือกกิจกรรมที่ 1 และให้ j = 1 แทนหมายเลขกิจกรรมที่เลือก
4. สำหรับ i = 2 ถึง n

   ถ้า $s_i >= f_j$ แล้ว เลือกกิจกรรมที่ i และให้ j = i

เช่น หลังจากเรียงข้อมูลตามเวลาสิ้นสุดจากน้อยไปมากจะได้ ดังนี้

start → finished

i	s_i	f_i	
1	1	4	*
2	3	5	
3	0	6	
4	5	7	*
5	3	8	
6	5	9	
7	6	10	
8	8	11	*
9	8	12	
10	2	13	
11	12	14	*

กิจกรรมที่มีเครื่องหมาย * อยู่ข้างหลัง คือ กิจกรรมที่ถูกเลือก

Job Sequencing Problem

กำหนดอาร์เรย์ของงานให้ โดยทุก ๆ งานมี deadline และกำไรที่ได้เมื่องานนั้นทำเสร็จสิ้น ก่อน deadline ที่กำหนดไว้ข้างต้น **งานที่ให้แต่ละงานใช้เวลา 1 หน่วย** ดังนั้นค่าน้อยสุดของ deadline ที่กำหนดให้คือ 1 สามารถหาผลรวมของกำไรสูงสุดของงานที่ได้จัดลำดับตามข้อกำหนดดังกล่าวได้ดังนี้

ตัวอย่างที่ 1 ประกอบด้วยงาน 4 งานแสดง deadline และ

JobID	Deadline	Profit
a	4	20
b	1	10
c	1	40
d	1	30

ผลรวมของกำไรสูงสุดของงานที่ได้จัดลำดับตามข้อกำหนดคือ 60
โดยงานที่ถูกนำมาทำตามลำดับคือ c, a

ตัวอย่างที่ 2 ประกอบด้วยงาน 5 งานแสดง deadline และ

JobID	Deadline	Profit
a	2	100
b	1	19
c	2	27
d	1	25
e	3	15

ผลรวม = 25 + 15 + 100
= 140

ผลรวมของกำไรสูงสุดของงานที่ได้จัดลำดับตามข้อกำหนดคือ
โดยงานที่ถูกนำมาทำตามลำดับคือ

Algorithm :

ระบบปฏิบัติการ: CPU Scheduling

กำหนดให้ CPU หนึ่งตัว สามารถรัน process ได้ทีละ process และมี process เข้ามาพร้อมกันหลาย process ในเวลาที่ 0 แต่ละ process ใช้เวลาต่างกัน (CPU burst time)

คำถาม: ต้องจัดลำดับให้ CPU รัน process ใดก่อนหลัง ลำดับอย่างไร จึงจะมี **average waiting time** ต่ำที่สุด โดยที่

- Waiting time ของ process หนึ่ง คือเวลาที่ process นั้นต้องรอก่อนที่จะได้เริ่มรันบน CPU
- ไม่รวมเวลาที่ process รันบน CPU เช่น process P1 ได้เริ่มรันตอน 5 ms หลังจากเข้ามารอ และใช้เวลารันจนเสร็จ 2 ms จะนับว่า waiting time คือ 5 ms
- Average waiting time คือ average ของ waiting time ของทุก process

ปัญหาข้างต้น เรียกว่า Shortest-Job-First (SJF) เป็นอัลกอริทึมสาหรับ CPU Scheduling โดยเลือกรัน process ที่ burst time ต่าที่สุดก่อน ถ้าเสมอ เลือกอันไหนก็ได้ เป็น Greedy Optimal โดยให้ average waiting time ที่ต่ำที่สุด

ตัวอย่าง

มี 3 process

P1 มี burst time = 5 ms

P2 มี burst time = 1 ms

P3 มี burst time = 2 ms

- SJF: ทำ P2 -> P3 -> P1
- Waiting time ของ P1 = 3 ms
- Waiting time ของ P2 = 0 ms
- Waiting time ของ P3 = 1 ms
- Average waiting time = 4/3 ms

Greedy Algorithms for Special Cases of DP problems:

1. Minimum number of coins required
2. Fractional Knapsack Problem

ปัญหาแคชเชียร์ทอนเงิน

แคชเชียร์มีเหรียญ 1 บาท, 2 บาท, 5 บาท, และ 10 บาท จำนวนไม่จำกัด (ไม่มีธนบัตร) ต้องการทอนเงิน W บาท โดยใช้จำนวนเหรียญในการทอนน้อยที่สุด ต้องทอนอย่างไร

วิธีการโดยทั่วไป

1. ทอนเหรียญ 10 บาทให้มากที่สุด โดยไม่เกินจำนวนเงินที่ต้องทอน
2. หลังจากนั้น ทอนจำนวนเงินที่เหลือด้วยเหรียญ 5 ให้มากที่สุด โดยไม่เกินจำนวนเงินที่ทอน
3. ทำเหมือนเดิมกับเหรียญ 2 บาทและ 1 บาท

ตัวอย่าง W = 18

เหรียญ 10	เหรียญ 5	เหรียญ 2	เหรียญ 1	จำนวนเหรียญทั้งหมด
1	1	1	1	4

วิธีนี้เป็นวิธี Greedy เพราะการทอนเหรียญ 10 บาทให้มากที่สุดโดยไม่เกินจำนวนที่ต้องทอน เป็นสิ่งที่ดูเหมือนดีที่สุดในขณะนั้น (โดยไม่คำนึงถึงเงินที่เหลือที่ต้องทอนด้วยเหรียญอื่น)

พิสูจน์ว่า Greedy ได้คำตอบ Optimal สำหรับเงินไทย

สมมติว่ามีเหรียญ 1 บาท, 2 บาท, 5 บาท, และ 10 บาท (ไม่รวมธนบัตร)

ข้อสังเกตของ optimal solution ใด ๆ ก็ตาม

- จะใช้เหรียญบาทไม่เกิน 1 เหรียญเสมอ
- จะใช้เหรียญ 2 บาทไม่เกิน 2 เหรียญเสมอ
- จะใช้เหรียญ 5 บาทไม่เกิน 1 เหรียญเสมอ
- จำนวนเงินจากเหรียญ 1, 2, 5 บาท รวมกันไม่เกิน 10 บาท

optimal หรือไม่ .....

กรณี เพิ่มเหรียญ 4 บาท วิธีนี้ optimal หรือไม่.....

ตัวอย่าง ต้องทอนเงิน 8 บาท

เหรียญ 10	เหรียญ 5		เหรียญ 2	เหรียญ 1	จำนวนเหรียญทั้งหมด

เหรียญ 10	เหรียญ 5	เหรียญ 4	เหรียญ 2	เหรียญ 1	จำนวนเหรียญทั้งหมด

Running time ของวิธีการทอนเงินแบบ Greedy คือ O(n) เมื่อ n = จำนวนประเภทเหรียญที่มี (ถ้าไม่มีการเรียงมูลค่าเหรียญ)

ปัญหา Fractional Knapsack

ขโมยมีถุงใบหนึ่ง ที่รับน้ำหนักได้เต็มที่ W มีกองของมีค่าอยู่หลายประเภท โดยกองที่ i มีค่า v_i และมีน้ำหนัก w_i ขโมยต้องการตัดสินใจว่าจะเอาของมีค่าจากกองไหน กองละเท่าไหร่บ้างใส่ถุง โดยที่ถุงไม่ขาด และได้มูลค่ามากที่สุด ถ้าเลือกหยิบแค่น้ำหนัก x จากกอง i ให้ถือว่ามีค่า v_i*x/w_i เช่น ของที่มีค่า 4 และหนัก 2 ถ้าเลือกแค่น้ำหนัก 1 จะได้มูลค่า 2

ตัวอย่าง กำหนดให้ W = 100

W_i	W_1 = 1000	W_2 = 100
V_i	V_1 = 200	V_2 = 100

Greedy แบบที่ 1

Greedy แบบที่ 2

แต่ Greedy ไม่สามารถใช้กับ Knapsack problem ได้ เมื่อกำหนดให้ของเป็นชิ้น เช่น

มีของ 3 ชิ้น ถุงใส่ได้ W = 10

v1 = 8, w1 = 6, v1/w1 = 4/3

v2 = 5, w2 = 5, v2/w2 = 1

v3 = 4, w3 = 5, v3/w3 = 4/5

ดังนั้น greedy เลือกของชิ้น 1 ก่อน แล้วใส่ไม่ได้แล้ว มูลค่าทั้งหมด = 8

Optimal solution คือ ไม่เลือกชิ้น 1 เลือกชิ้น 2 กับชิ้น 3 มูลค่าทั้งหมด = 9 ซึ่งมากกว่าวิธี Greedy

ใช้ Dynamic Programming สำหรับปัญหา Knapsack problem

☺ DP is coming soon !!! ☺

แบบฝึกหัด Greedy Algorithm

1. ให้นักเรียนเขียนโปรแกรมเพื่อแก้ปัญหาการทอนเงิน โดยแคชเชียร์มีเหรียญ 1 บาท, 2 บาท, 5 บาท, และ 10 บาท จำนวนไม่จำกัด (ไม่มีธนบัตร) ต้องการทอนเงิน W บาท โดยใช้จำนวนเหรียญในการทอนน้อยที่สุด ต้องทอนอย่างไร

   ข้อมูลนำเข้า w แทนจำนวนเงินที่ต้องทอน

   ข้อมูลนำออก แสดงจำนวนเหรียญแต่ละประเภทตามลำดับ คือ เหรียญ 10 บาท, 5 บาท, 2 บาท, และ 1 บาท

   ตัวอย่าง

Input	Output
28	2 1 1 1
49	4 1 2 0

2. ให้นักเรียนเขียนโปรแกรมเพื่อหาจำนวนกิจกรรม โดยกำหนดให้มีห้องหนึ่งห้อง มีกิจกรรมที่จะต้องใช้ห้องหลายกิจกรรม แต่ละกิจกรรมจะมีเวลาเริ่มต้น และเวลาสิ้นสุด ต้องการเลือกกิจกรรมให้ได้มากที่สุด โดยแต่ละกิจกรรมที่ถูกเลือก ต้องมีเวลาไม่ทับซ้อนกัน ยกเว้นเวลาสิ้นสุดและเวลาเริ่มต้นของกิจกรรมต่อไป สามารถเป็นเวลาเดียวกันได้

   ข้อมูลนำเข้า บรรทัดที่ 1 n แทนจำนวนกิจกรรมที่มีทั้งหมด

   บรรทัดที่ 2 ถึงบรรทัดที่ n+1 ประกอบด้วยจำนวนเต็ม 2 จำนวน (s, f)

   จำนวนแรก s_i แทนเวลาเริ่มต้นของกิจกรรมที่ i

   จำนวนแรก f_i แทนเวลาสิ้นสุดของกิจกรรมที่ i

   ข้อมูลนำออก บรรทัดที่ 1 N แสดงจำนวนกิจกรรมที่มากที่สุดที่เลือกได้

   บรรทัดที่ 2 ถึงบรรทัดที่ N+1 แสดงรายการกิจกรรมที่ถูกเลือก

ตัวอย่าง

Input	Output
11 0 6 1 4 2 13 3 5 3 8 5 7 5 9 6 10 8 11 8 12 12 14	4 1 4 5 7 8 11 12 14

3. ให้นักเรียนเขียนโปรแกรมเพื่อแก้ปัญหา fractional knapsack

ข้อมูลนำเข้า บรรทัดที่ 1 W แทนน้ำหนักที่ถุงจะรับได้มากที่สุด
N แทนจำนวนกอง
บรรทัดที่ 2 ถึงบรรทัดที่ n+1 ประกอบด้วยจำนวนเต็ม 2 จำนวน (v, w)
จำนวนแรก v_i แทนมูลค่าของกองที่ i
จำนวนแรก w_i แทนน้ำหนักของกองที่ i

ข้อมูลนำออก แสดงมูลค่ามากที่สุดของของในถุง

ตัวอย่าง

Input	Output
100 2 200 1000 100 100	100.00
4 3 100 2 10 2 120 3	180.00

ตารางเรียนออนไลน์ (Course Schedule)

งานที่ต้องทำคือ จัดตารางเรียนออนไลน์ โดยมีคอร์สเรียนออนไลน์ทั้งหมด n คอร์ส (1 ถึง n) แต่ละคอร์สมีระยะเวลาสอน t และจะปิดคอร์สในวันที่ d^{th} คอร์สดังกล่าวใช้วันสอนต่อเนื่องตามระยะเวลาที่กำหนด สำหรับ t วัน การจัดคอร์สลงตารางเรียน คอร์สแรกที่เลือกเริ่มต้นที่วันที่ 1 และในช่วงเวลาเดียวกันจะไม่มีการจัดคอร์สซ้ำซ้อนในเวลาเดียวกัน ให้นักเรียนเขียนโปรแกรมเพื่อหาจำนวนคอร์สที่มากที่สุดที่สามารถจัดลงตารางตามเงื่อนไขที่กำหนดได้

ข้อมูลนำเข้า บรรทัดที่ 1 จำนวนเต็มสองจำนวน n แทนจำนวนคอร์ส โดยที่ $2 \leq n \leq 100$

บรรทัดที่ 2 ถึงบรรทัดที่ $n+1$ จำนวนเต็ม t และ d แทนจำนวนวัน (ระยะเวลาที่ใช้ของคอร์สที่ i) และวันที่ ที่คอร์สต้องสิ้นสุดก่อน

โดยที่ $1 \leq t \leq 1000$ และ $1 \leq d \leq 10000$

ข้อมูลนำออก จำนวนคอร์สที่มากที่สุดที่สามารถจัดลงตารางตามเงื่อนไขที่กำหนดได้

ตัวอย่าง

Input	Output
4 100 200 200 1300 1000 1250 2000 3200	3
3 2 6 3 3 4 4	2
3 5 5 4 6 2 6	1

ตัวอย่างที่ 1 คอร์สที่ถูกเลือกคือ [100 200] อันดับแรก ถัดมาคือ [1000 1250] โดยเริ่มทำงานในวันที่ 101 เนื่องจากคอร์สแรกที่เลือกใช้เวลา 100 วัน และสิ้นสุดในวันที่ 1001 คอร์สที่ 3 ที่เลือกคือ [200 1300] สิ้นสุดในวันที่ 1300 ส่วนคอร์ส [2000 3200] ไม่ถูกเลือกเนื่องจากใช้เวลา 2000 วัน ทำให้สิ้นสุดวันที่ 3300 เกินวันสิ้นสุดที่กำหนด คือ วันที่ 3200 ดังนั้นสามารถเลือกคอร์สจัดลงตารางได้มากที่สุด เท่ากับ 3

ตัวอย่างที่ 2 เลือกได้มากสุด 2 คอร์ส คือแบบที่หนึ่ง [3 3] และ [2 6] หรือแบบที่สอง [4 4] และ [2 6]

ตัวอย่างที่ 3 เลือกได้มากสุด 1 คอร์ส

การจัดงานที่เป็นลำดับ (Job Sequencing Problem)

กำหนดอาร์เรย์ของงานมาให้ โดยที่ทุก ๆ งานจะมี deadline และกำไร (profit) จากงานนั้น ๆ ถ้างานนั้นทำเสร็จต้องการจัดงานให้เป็นไปตามกำหนดเวลา โดยได้กำไรที่มากที่สุด

ตัวอย่าง Input: Four Jobs with following deadlines and profits

JobID	Deadline	Profit
a	4	20
b	1	10
c	1	40
d	1	30

Output: Following is maximum profit sequence of jobs

c, a

Input: Five Jobs with following deadlines and profits

JobID	Deadline	Profit
a	2	100
b	1	19
c	2	27
d	1	25
e	3	15

ควรปรับให้ output เป็น profit ที่ได้มากที่สุด เนื่องจากลำดับงานไม่ fix ให้สามารถแสดง a หรือ c ก่อนก็ได้

Output: Following is maximum profit sequence of jobs

c, a, e

ให้นักเรียนเขียนโปรแกรมเพื่อแก้ปัญหาการจัดงานที่เป็นลำดับ (Job Sequencing Problem)

Input	Output
4 a 4 20 b 1 10 c 1 40 d 1 30	c a
5 a 2 100 b 1 19 c 2 27 d 1 25 e 3 15	c a e

Minimum Cabs

ให้ผู้โดยสารจำนวน N คน และผู้โดยสารแต่ละคนต้องการรถแท็กซี่ 1 คัน สำหรับแต่ละคนจะกำหนดเวลาเริ่มต้น (start time) และเวลาสิ้นสุด (end time) สำหรับการเดินทาง ให้นักเรียนเขียนโปรแกรมเพื่อหาจำนวนรถแท็กซี่ที่น้อยที่สุดที่ต้องการสำหรับคนกลุ่มนี้

ข้อมูลนำเข้า บรรทัดที่ 1 ประกอบด้วยจำนวนเต็ม N โดยที่ $1 \leq N \leq 10^5$
บรรทัดที่ 2 ถึงบรรทัดที่ $N+1$ ประกอบด้วยจำนวนเต็ม 4 จำนวน $HH1\ \ MM1\ HH2$ และ $MM2$ โดยที่ $(0 \leq HH1,\ HH2 \leq 23)\ (0 \leq MM1,\ MM2 \leq 59)$
รับประกันว่าข้อมูลเวลาดังกล่าวไม่เป็นเที่ยงคืน

ข้อมูลนำออก แสดงจำนวนรถแท็กซี่ที่น้อยที่สุดที่ต้องการ

Input	Output
6 1 0 2 0 16 0 21 30 9 30 13 0 21 30 22 30 21 30 22 30 12 0 12 30	3

คำอธิบายเพิ่มเติม

$N = 6$

1. 01:00 – 02:00 ผู้โดยสารคนที่ 1 ใช้รถคันแรก
2. 16:00 – 21:30 ผู้โดยสารคนที่ 2 ใช้รถคันแรก เพราะรถคันแรกว่างในช่วงเวลานี้
3. 09:30 – 13:00 ผู้โดยสารคนที่ 3 ใช้รถคันแรก
4. 21:30 – 22:30 ผู้โดยสารคนที่ 4 ใช้รถคันที่ 2
5. 21:30 – 22:30 ผู้โดยสารคนที่ 5 ใช้รถคันที่ 3
6. 12:00 – 12:30 ผู้โดยสารคนที่ 6 ใช้รถคันที่ 2 เนื่องจากคนที่ 3 ใช้รถคันแรกในช่วงเวลานี้

ดังนั้นต้องใช้รถทั้งหมด 3 คัน

Rooms

เจ้าของโรงแรมแห่งหนึ่งต้องการคำนวณหาจำนวนห้องพักที่น้อยที่สุดที่ต้องสร้าง โดยเขามีรายการการเข้าพักของลูกค้าประกอบด้วย วันที่มาถึง และจำนวนวันที่ลูกค้าต้องการพัก ทั้งนี้เขาจะจัดห้องให้ลูกค้าหนึ่งคนพักหนึ่งห้องเพื่อให้ลูกค้าพึงพอใจ

ข้อมูลนำเข้า บรรทัดที่ 1 ประกอบด้วยจำนวนเต็ม T แทนจำนวนชุดทดสอบ โดยที่ $1 \leq T \leq 5$ สำหรับแต่ละชุดทดสอบ

บรรทัดที่ 1 จำนวนเต็ม N แทนจำนวนลูกค้าที่ต้องการเข้าพัก โดยที่ $1 \leq N \leq 10^5$

บรรทัดที่ 2 จำนวนเต็ม N จำนวนคั่นด้วยช่องว่าง 1 ช่อง แทนวันที่ลูกค้ามาถึง

บรรทัดที่ 3 จำนวนเต็ม N จำนวนคั่นด้วยช่องว่าง 1 ช่อง แทนจำนวนวันที่ลูกค้าต้องการพัก

ข้อมูลนำออก จำนวนห้องพักที่น้อยที่สุดที่เจ้าของโรงแรมต้องสร้าง

ตัวอย่าง

Input	Output
2 3 1 2 3 3 3 3 5 1 2 3 4 5 2 3 4 5 6	3 3

เศษส่วนอียิปต์ (Egyptian Fraction)

เศษส่วนที่เป็นบวกทุก ๆ จำนวนที่สามารถเขียนแทนในรูปของผลบวกของเศษส่วนโดยเศษมีค่าเท่ากับ 1 เท่านั้น (sum of unique unit fractions) เรียกเศษส่วนดังกล่าวว่า เศษส่วนอียิปต์ (Egyptian Fraction)

ตัวอย่าง

เศษส่วนอียิปต์ 2/3 สามารถแทนด้วย 1/2 + 1/6

เศษส่วนอียิปต์ 6/14 สามารถแทนด้วย 1/3 + 1/11 + 1/231

เศษส่วนอียิปต์ 12/13 สามารถแทนด้วย 1/2 + 1/3 + 1/12 + 1/156

เราสามารถเขียนเศษส่วนอียิปต์โดยใช้ Greedy Algorithm ได้ดังนี้

สำหรับ 6/14

- หา ceiling ของ 14/6 ได้ 3 ดังนั้นเศษส่วนแรก คือ 1/3
- ทำซ้ำสำหรับ (6/14 – 1/3) จนได้ผลรวมทั้งหมดเท่ากับ 6/14 นั่นคือ ส่วน หาร เศษ เท่ากับ 0

ข้อมูลนำเข้า a, b เป็นจำนวนเต็ม 2 จำนวน แทนเศษและส่วน คั่นด้วยช่องว่างหนึ่งช่อง

ข้อมูลนำออก แสดงเลขที่เป็นส่วนทั้งหมดของเศษส่วนอียิปต์

ตัวอย่าง

Input	Output
2 3	2 6
6 14	3 11 231
12 13	2 3 12 156

Ferry Loading

เรือเฟอร์รี่สำหรับขนรถยนต์ข้ามฟาก สามารถบรรทุกรถยนต์ได้ n คัน โดยใช้เวลาขาไปและขากลับเที่ยวละ t นาที ให้นักเรียนคำนวณหาเวลาที่น้อยที่สุดสำหรับขนรถยนต์ข้ามฟากและจำนวนเที่ยวที่ใช้

ข้อมูลนำเข้า บรรทัดที่ 1 ประกอบด้วยจำนวนเต็ม T แทนจำนวนชุดทดสอบ โดยที่ $1 \leq T \leq 5$ สำหรับแต่ละชุดทดสอบ

บรรทัดที่ 2 จำนวนเต็ม n, t และ m โดยที่ n แทนจำนวนรถยนต์ที่เรือเฟอร์รี่สามารถบรรทุกได้ t แทนเวลาที่เรือเฟอร์รี่ใช้สำหรับข้ามฟากมีหน่วยเป็นนาที และ m แทนจำนวนรถยนต์ที่มารอข้ามฟาก

บรรทัดที่ 3 ถึงบรรทัดที่ $n + 3$ แสดงเวลาที่รถยนต์แต่ละคันมาถึงท่าเรือ

ข้อมูลนำออก ประกอบด้วยจำนวนเต็มสองจำนวน คือ เวลาที่น้อยที่สุดสำหรับขนรถยนต์ข้ามฟากและจำนวนเที่ยวไปกลับที่เรือเฟอร์รี่ใช้ในการขนรถยนต์ทั้งหมดที่กำหนดให้

ตัวอย่าง

Input	Output
2 2 10 10 0 10 20 30 40 50 60 70 80 90 2 10 3 10 30 40	100 5 50 2

ของเล่น (Toys)

มาร์กเป็นคุณพ่อที่ต้องการซื้อของเล่นให้ลูก โดยมีหลักเกณฑ์ในการซื้อของเล่นคือ ใช้เงินที่มีซื้อของเล่นให้ได้จำนวนชิ้นมากที่สุด มาร์กได้รับ รายการราคาของเล่นและจำนวนเงินงบประมาณ

ตัวอย่าง

กำหนดรายการราคาของเล่น $prices = [1, 2, 3, 4]$ และจำนวนเงินงบประมาณ $k = 7$

มาร์กเลือกซื้อได้สองแบบ คือ $[1, 2, 3]$ และ $[3, 4]$ สำหรับเงิน 7 บาท ดังนั้นมาร์กจะแบบแรกเพราะได้ของเล่น 3 ชิ้น

ข้อมูลนำเข้า บรรทัดที่ 1 จำนวนเต็มสองจำนวน n และ k แทนจำนวนของเล่นและเงินงบประมาณตามลำดับ โดยที่ $2 \leq n \leq 100$ และ $2 \leq k \leq 100000$

บรรทัดที่ 2 รายการราคาของเล่น p_i โดยที่ $(1 \leq i \leq n)$ แทนด้วยจำนวนเต็ม n จำนวนคั่นด้วยช่องว่าง โดยที่ $2 \leq p_i \leq 1000000$

ข้อมูลนำออก แสดงจำนวนของเล่นที่ซื้อได้มากที่สุดจากเงินงบประมาณที่กำหนดให้

ตัวอย่าง

Input	Output
7 50 1 12 5 111 200 1000 10	4

มาร์กสามารถซื้อได้ 4 ชิ้น จาก 1 12 5 10

คู่ (Pairs)

กำหนดอาร์เรย์ของจำนวนเต็มและค่าเป้าหมาย ให้หาจำนวนคู่ผลต่างของสมาชิกในอาร์เรย์ซึ่งมีค่าเท่ากับค่าเป้าหมาย

ตัวอย่าง

อาร์เรย์ของจำนวนเต็ม $[1, 2, 3, 4]$ และค่าเป้าหมาย เท่ากับ 1

จำนวนคู่ของสมาชิกในอาร์เรย์ที่ให้ผลต่างเท่ากับ 1 มี 3 คู่ คือ $[2, 1], [3,2], [4, 3]$

ข้อมูลนำเข้า บรรทัดที่ 1 จำนวนเต็มสองจำนวน n และ k แทนจำนวนสมาชิกในอาร์เรย์และค่าเป้าหมายตามลำดับ

บรรทัดที่ 2 จำนวนเต็ม n จำนวน คั่นด้วยช่องว่าง

ข้อมูลนำออก แสดงจำนวนคู่ผลต่างของสมาชิกในอาร์เรย์ที่มีค่าเท่ากับค่าเป้าหมายที่กำหนด

ตัวอย่าง

Input	Output
5 2 1 5 3 4 2	3

จำนวนคู่ของสมาชิกในอาร์เรย์ที่ให้ผลต่างเท่ากับ 2 มี 3 คู่ คือ [5,3], [4,2], [3,1]

Max Min

กำหนดให้ arr เป็นอาร์เรย์ของจำนวนเต็ม และ k เป็นจำนวนเต็ม ให้นักเรียนหาผลต่างที่น้อยที่สุดของ$subarr$ อาร์เรย์ขนาด k โดยอาร์เรย์นี้สร้างจากสมาชิกของ arr ที่กำหนดให้

สามารถคำนวณ $max(subarr) - min(subarr)$

โดยที่ max คือค่าที่มากที่สุดใน $subarr$

min คือค่าที่น้อยสุดใน $subarr$

ข้อมูลนำเข้า บรรทัดที่ 1 จำนวนเต็มสองจำนวน n และ k แทนจำนวนสมาชิกในอาร์เรย์และค่าเป้าหมายตามลำดับ

บรรทัดที่ 2 ถึงบรรทัดที่ $n + 1$ จำนวนเต็ม n จำนวน

ข้อมูลนำออก แสดงผลต่างที่น้อยที่สุดของ $subarr$ อาร์เรย์ขนาด k

ตัวอย่าง

Input	Output
7 3 10 100 300 200 1000 20 30	20
10 4 1 2 3 4 10 20 30 40 100 200	3
5 2 1 2 1 2 1	0

Greedy for Water

ให้นักเรียนเขียนโปรแกรมเพื่อหาจำนวนขวดที่มากที่สุด สำหรับการบรรจุน้ำ โดยกำหนดปริมาณน้ำและขวดขนาดต่าง ๆ ให้

ตัวอย่าง

ข้อมูลนำเข้า บรรทัดที่ 1 จำนวนเต็มสองจำนวน n และ v แทนจำนวนขวดน้ำที่มีและปริมาณน้ำที่กำหนดให้ตามลำดับ

บรรทัดที่ 2 จำนวนเต็ม n จำนวน แทนปริมาณน้ำที่แต่ละขวดสามารถบรรจุได้

ข้อมูลนำออก แสดงจำนวนขวดที่มากที่สุดที่สามารถบรรจุน้ำตามที่กำหนดได้

ตัวอย่าง

Input	Output
5 10 8 5 4 3 2	3